A Model for Autonomous Reconfiguration of Earth Sensing Resources

Robert Morris[1], Jennifer Dungan[1], Petr Votava[1,2] Lina Khatib[1,3]
(1) NASA Ames Research Center
(2) California State University, Monterey Bay
(3) Perot Systems Government Services

***Abstract*— An Earth-observing sensor web is an organization of space, airborne, or in situ sensing devices for collecting measurements of the Earth's processes. Potential users of a sensor web include Earth scientists and authorities responsible for disaster monitoring, mitigation and management. Effective utilization of the collection of sensing and data processing resources requires a coordinated infrastructure to access them and the data they produce. The focus of this paper is to describe in detail a high level theory and model of sensor web coordination. The basis of this approach is the idea that a sensor web can be viewed as a complex *controllable* physical system. From this viewpoint it is then possible to leverage recent advances in *autonomous control* technology to automate the process of accomplishing Earth science goals by *reconfiguring* the resources for acquiring, storing and analyzing data.**

I. Introduction

An Earth science *sensor web* is a distributed organization of sensors, Earth science models, human scientists and information technologists, and data archives. The sensors in the web will typically vary with respect to the type of observations they make, and with respect to the platform on which they reside: the platform can be fixed or mobile, ground-based, airborne, or space-borne, etc. An Earth science sensor web is one of a number of different types of sensor webs configured to assist in human understanding: other sensor webs arise in military contexts (e.g. in battlefield management) and in disaster mitigation (e.g. in hurricane tracking).

The scientific use of sensor web consists broadly of seeking to improve the understanding of natural processes occurring on the Earth's surface or in the atmosphere. Sensor measurements serve to quantify aspects of these processes that allow Earth science models to make predictions of scientific and social value. Other measurements are intended to improve the cohesion or consistency of the data of the sensor web itself: for example, measurements from one sensor can be used to validate observations taken by another.

The sensor webs control problem to be considered here is the problem of *reconfiguration*. Sensor web reconfiguration is the process of generating and executing a set of actions that will re-organize a subset of the set of resources of the web. Currently, reconfiguring a sensor web is primarily conducted by humans, with varying degrees of automation. Human expertise for sensor web reconfiguration comes in three broad types: *scientific* expertise to formulate requirements for observation; *data management* expertise to collect the data and set up and execute processing tasks; and *mission planning* expertise for setting up and scheduling the sensing platforms to take measurements.

Much of the reconfiguration tasks performed by humans is tedious and time-consuming, and could be more effectively performed by machines. Recent advances in software auutomation can be applied to improving sensor web management by

- Making the web more *goal-directed*. If reconfiguring is directly tailored to specific science goals, the data acquired will be of more scientific value.
- Promoting *abstraction*. Humans can formulate goals at a high level, and the automated system will automatically work out the details in how to accomplish them.
- Making the web more *robust*. Unexpected changes to the sensor web that hinder its ability to be reconfigured in a planned way can be diagnosed and an alternative configuration that satisfies the same goals can be generated automatically.
- Improving web *effectiveness*. Better reconfiguration will mean enhanced capability for information flow, thus improving the turnaround time for goal achievement.
- Improving web *efficiency*. Because of the complexity of manually reconfiguring a distributed system, most sensor web control tends to be based on the strategy that can be roughly summarized as *observe everything and sort it out later*. This strategy is wasteful of resources and ultimately untenable as the size of the sensor web grows.

The purpose of this paper is to introduce a model for sensor web reconfiguration. The model employs concepts underlying finite state machines to isolate the principles underlying goal-directed reconfiguration. This model forms the basis for greater automation of the reconfiguration process.

The remainder of this paper is structured as follows: Section 2 introduces the stages of an *observation cycle*. Each component in the cycle is then examined in detail, leading to the formulation of the finite state model of reconfiguration. The paper ends with a discussion of an implementation of the model currently being developed.

II. The Cycle of Sensor Web Reconfiguration: an Example

A high-level summary of the sensor web observation cycle is displayed in Figure 1. Labels on the arcs indicated phases

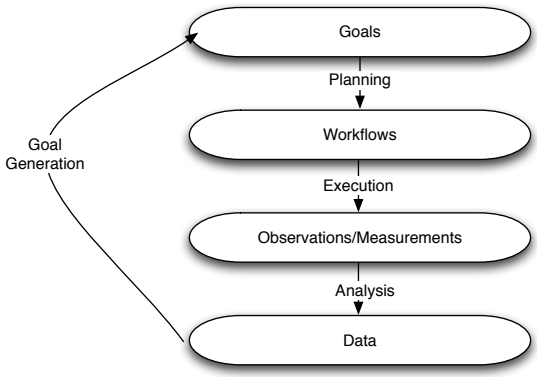

Fig. 1. Phases in the Data Acquisition Cycle.

in the cycle, and the labels on the ovals indicate the results of each phase. Four phases in the cycle are distinguished: goal generation, workflow planning, workflow execution and analysis. This cycle is somewhat overly simple, insofar as it could contain other loops involving a subset of the phases. For example, not all data are transformed into new goals; rather some could be used to modify an existing workflow.

Here is a real example to illustrate each stage in an observation cycle. INTEX-NA [7] was an integrated atmospheric field experiment performed over North America that occurred for ten weeks between March and May, 2006. Its overall objective was to understand the transport and transformation of gases and aerosols on transcontinental/intercontinental scales and their impact on air quality and climate. The main measurements of interest were ozone, aerosols, and greenhouse gasses. The "sensor web" for INTEX-NA consisted of sensors mounted in airborne platforms on a DC-8 and P-3B, as well as satellite measurements from Terra (MODIS, MISR, MOPITT), Envisat (SCIAMACHY, MIPAS), and Aura (TES, OMI), and another airborne platform, the NSF/NCAR C-130.

The INTEX-NA observation cycle occurred each day during the experiment. Out of the overall mission objectives each day came a set of *observation goals*. For example, a goal might be stated as *sample aged Mexico City pollution outflow and its nearsource characterization using both remote and in situ measurements*. To satisfy this goal, it was necessary to reconfigure the sensor web to retrieve data from the remote sensing instrument and retarget the DC-8 to fly a certain pattern. The actions that accomplish this reconfiguration is the specific *workflow* for this day. Roughly, a workflow is a sequence of actions (a plan) for accomplishing the goal. The *execution* of the workflow consists of the process of obtaining the in situ measurements taken by DC-8 (i.e. the flight itself), as well as the retrieval of the remote sensing data. An example of a *measurement* that results from a workflow is *at 25,000 feet tropical air influences were encountered with low ozone (18 ppb) and high humidity*. Such observations are often analyzed with respect to predictions made by models. What we mean broadly here by *data* is the result of some *processing* action. A simple example would be to process the error between the predicted concentrations of CO2 at a given altitude and location and the observed concentration. The (average) error are the data that results from the analysis. Such data often indicate new goals for the next cycle.

The boundary between "observations" and "data" is notoriously fuzzy, but fortunately it is not necessary for the purposes of this paper to make the boundary crisp. For our purposes, "data" can mean anything that provides useful information for initiating the next observation cycle, or, more generally, that contributes to the overall accomplishment of goals.

The next sections drill deeper into a characterization of each of the phases of the observation cycle.

III. Goal generation

The observation cycle begins with one or more *science goals*. Goals are high-level specifications of required data products for the science investigation. Science goals commonly arise from the incremental development and refinement of models for simulating and predicting Earth processes. Types of goals include

- *Characterize/classify*: obtain or identify the values of a quantity of interest. Example: *Characterize the chemical aging of Asian pollution plumes during travel across the Pacific* [7].
- *Monitor*: watch for a significant change or for the threshold of some quantity to be reached. *Example: monitor carbon dioxide uptake and release at the Earth's surface over time.*
- *Compare*: determine the similarities and/or differences between or among several instances of the same process or quantity. *Example: Compare the average soil moisture for a specified region over the next 6 months with 30 year average.*
- *Validate*: determine whether a prediction made by a model is correct or whether observations from a sensor are accurate. *Example: Validate the numerical model simulation of the global hydrological cycle using tropical rainfall data.*
- *Predict*: determine when a specific event will happen or how a quantity will evolve in the future. *Example: apply expected precipitation data to measurements of the initial conditions of soil moisture and river flows to simulate headwater rainfall partitioning for predicting the routing of flood waves.*

A goal can be viewed as a feature vector containing, but not limited to, the following attributes:

- *what* is to be measured (e.g. CO2);
- *where* the measurement(s) are to be taken;
- *when* the measurement(s) are to be taken; and
- *how* the data are acquired, i.e., the instrument or sensor employed.

The values of these attributes may be obtained in different ways. First, they may be assigned *statically* (at planning time, the time the goal is initially specified) or dynamically, i.e., during workflow execution, as the result of other actions

in the workflow). For example, the frequency with which a measurement is made may depend dynamically on something occurring in the world that is not known at planning time (e.g. the occurrence of a volcanic eruption).

Secondly, values of attributes may be completely user-specified, or may be specified through interaction with a *discovery service* [4]. To illustrate the difference, suppose a user requires soil moisture measurements from a given location and time, and also knows how to invoke the resource directly to acquire this data. Then, the user can completely specify all the needed parameters related to this goal. By contrast, suppose the user knows the location and time of the required measurement, but does not know the sensor that could provide the data, or even whether the resource exists. If a knowledge discovery service layer is available, then the user can invoke the service layer to provide the required sensor information. Then the user can complete the goal specification.

Figure 2 illustrates some of the possible interactions between the user and the sensor web, for the simple scenario in which the user makes a request for data, and a sensor resource controller replies with an acknowledgment. There are two intermediate layers, the *planner layer* and *service layer* that provide different automated capabilities to help the user. In the first case (top) the user interacts directly with the sensor web. In the second case (middle) the user utilizes the service layer for the purpose of discovering resources that can be used to acquire the needed data. In the third case (bottom), the user utilizes the planner layer to transform a high level goal into one or more requests. The planner layer, in turn, uses the service layer in this scenario for discovery purposes. Note that if the user is so inclined, the planner will relay the acknowledgment back to the user.

IV. Workflow generation

A goal, or set of goals, is decomposed into a sequence of actions, called a *workflow* that, when executed as planned, reconfigures the sensor web in a way that accomplishes the goal. In this section, an abstract model of workflows is proposed based on a framework that views the automated process of acquiring data as a set of states of a finite state machine. We use a graphical representation called a *Labeled Transition System*[1] (LTS) for depicting states and transitions. In this graphical representation, machine state transitions are modeled as labels consisting of names of actions. workflows are described in an algebraic variant of the LTS notation called *Finite State Processes* (FSP).

A. Actions and processes

There are two kinds of actions for performing web reconfiguration: *web access* (WA) actions, and *staging* actions. WA actions access the resources of the sensor web to achieve workflow goals. There are three atomic WA actions: *acquire data*, *process data*, and *store data*, reflecting the stages of requesting, processing, and retrieving data products.

Actions are organized as concurrent *processes*. Processes are composed by a number of ordering operations: sequencing (putting one action after another), choice (doing one action or another), iteration (repeating an action sequence) and parallelism (doing more than one action in parallel). A state of the workflow is defined as the combined state of each process in the workflow. Similarly. a state of a process is defined as the action being executed in the state by the process, as well as the values of local state variables.

1) Web Access Actions: By a *measurement*, we mean a pair consisting of a phenomenon of interest (measurement type) and an associated value. By an *observation*, we mean an event that results in a measurement, i.e. a binding of a quantity to a measurement type. The outcome of an acquire data action is a collection of measurements.

A process data action refers to any numerical or logical procedure that extracts information of interest from acquired data. An example of a processing action includes computing the average over a set of measurements. The result of a process data action is therefore a numerical value. Data process actions are typically repeated on different inputs.

A store data action produces a data product required by the workflow goals. Typically store actions consist of copying the output of a data processing action to a designated location for retrieval by the end user (the person issuing the initial request for the data).

2) Staging Actions: Staging actions interleave with WA actions within a process to control the progress of executing a workflow. A staging action is associated with a boolean expression called a *trigger* that must be true for the action to be executed (equivalently: for the state transition to be applied). A trigger may be a condition observed in the world, or the result of a calculation performed by the executing system. Finally, if two or more processes share the same (staging) action, then the execution of the action is not performed by one process unless it is performed by all. This interpretation allows for orderings among actions in different processes to occur, as will be evident shortly.

Staging actions are used for three purposes. First, as a mechanism for *branching*. Such staging actions are associated with two or more transitions in an LTS, where only one of the triggers associated with the transition can be true. Second, some staging actions are used for *synchronization*, which is a way to impose an order on a set of WA actions. For example, a processing WA action may process data from an acquisition WA action. To enforce the dependency of the processing action on its inputs, a synchronizing action is introduced. Finally, a staging action may be used to *wait* until some response from the sensor web is received. For example, a staging action can be used to wait for notification that a set of measurements have been taken.

Actions have parameters or *arguments*. The arguments of WA actions are as follows:

- acquire data action: indicators of the location and time of a measurement, and the sensing instrument, as well perhaps other parameters indicating the quality requirements of the acquisition;

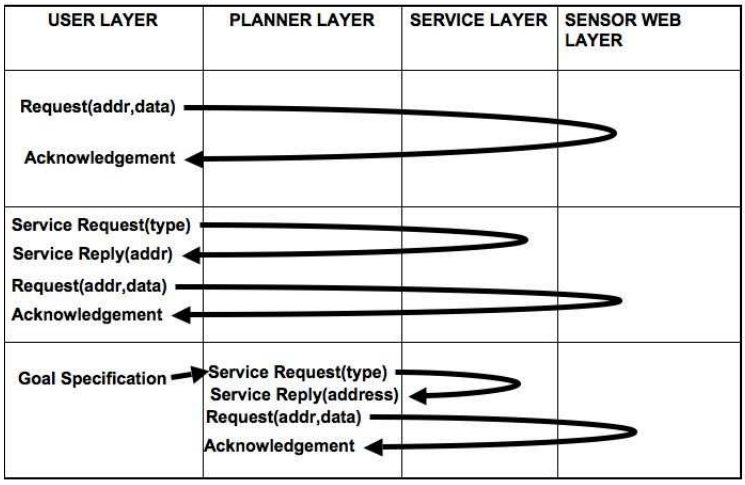

Fig. 2. Three scenarios illustrating the interactions of the user with different sensor web service and access layers. Top: user requests data directly from sensor web. Middle: user utilizes service layer to discover sensing resources. Bottom: user formulates goal and planner layer generates and executes data acquisition request, utilizing service layer. Adapted from [4].

- process data action: the nature of the processing to be done, the location of the function, and the required inputs;
- store action: an address (e.g. URL) indicating the location of a requested data product.

By contrast, the arguments of staging actions are the variables and values that make up the associated trigger. For example, as illustrated below, the *done* staging action is associated with a boolean expression that evaluates to $true$ if all the data have been acquired.

B. Process-based Abstract Workflows

As noted above, each WA action in a workflow is composed into a WA *process*, an LTS whose transitions consist of staging or data acquisition actions. [1] Figure 3 shows a LTS depiction of the stages of an workflow process. The actions *ready, not ready, done, not done* are branching staging actions. *Ready* tests whether data are available to be acquired (for example, if enough time has past for the scheduled measurements to have been taken). *Done* tests whether all the requested data have been acquired. get_acq1 is the WA action, and end_acq1 is a synchronizing staging action, which can be used when this acquisition is tied sequentially to some other action, such as a data processing action that requires the acquired data as inputs. The FSP version of the LTS transition network can be depicted as in Table I. For the sake of readability, we have broken the acquisition process into three sub-processes. Roughly, the first sub-process ($InitGET\_acq$) checks to ensure the preconditions of a a data acquisition tasks are true (represented by the staging action $ready$). The sub-process GET_acq contains the acquisition action itself. Finally, End_acq checks whether all the data have been acquired, via the staging action $done$.

This simple example illustrates the way processes are constructed. One fundamental notion is that of an *action prefix*: an expression of a form $act \rightarrow P$, where act is an action and P is a process. In addition, to express branching on the value of $ready$ or $done$, the vertical bar | is used. A special $STOP$ process is defined to terminate the entire process. Implicit in the FSP description are the arguments for each action. To reveal these arguments, we could write, for example, $ready(E)$ and $not_ready(\neg E)$where $E, \neg E$ are the associated triggers.

C. Process Synchronization and Parallelism

The processes of a workflow are executed in parallel. Orderings among the processes are accomplished through synchronization. As noted above, synchronizing staging actions must be executed at the same time by all the processes that participate in that action [1]. In FSP notation, the assertion

||PAR = (InitGET_acq | InitPROCESS | InitSTORE).

states that the three acquisition processes execute in parallel. Notice that all of the atomic WA actions in a workflow are organized as processes with similar three-part structures, consisting of a $ready$ test, the WA action, and the $done$ test. This structure enables an efficient process of automatically generating the workflow from the goal specification.

D. Conditional Actions

Imagine a scenario in which the goal specifier wants only cloud-free data. This can be modeled by a processing action followed by an intermediate process called a *test*, that acts like a filter of data. A test process consists of a branching staging

[1]Note the dual use of the term *process* here, which was unavoidable due to the desire to use standard terminology. To avoid ambiguity, we will use the distinct expressions *process associated with a workflow* and *data process.*

InitGET_acq = (ready → GET_acq | not_ready → Init_GET),
GET_acq = (get_acq1 → End_acq),
End_acq = (end_acq1 → (not_done → InitGET_acq | done → STOP))

TABLE I

A WORKFLOW AS A FINITE STATE PROCESS (FSP), COMPRISED OF WEB ACCESS ACTIONS AND STAGING ACTIONS FOR ACQUIRING DATA.

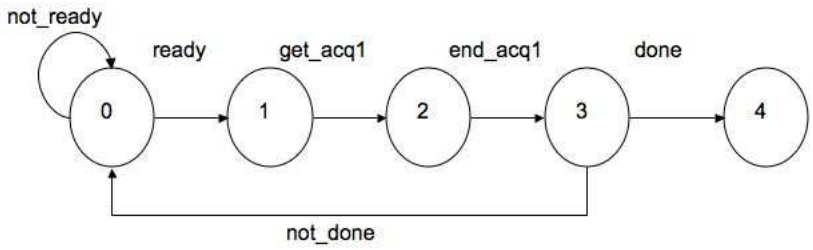

Fig. 3. The Stages of Data Acquisition as a Labeled Transition Network (LTS).

InitPROCESS_p1 = (end_acq → PROCESS_p1),
PROCESS_p1 = (process_p1 → WrapPROCESS_p1),
WrapPROCESS_p1 = (end_p1 (not_done1 → InitPROCESS_p1 | done1 → STOP)).

InitSTORE_s1= (end_p1 → TEST_t1),
TEST_t1 = (yes → STORE_s1 | no → WrapSTORE),
STORE_s1 = (store_s1 → WrapSTORE),
WrapSTORE =(not_done1 → InitSTORE_s1 | done1 → STOP).

TABLE II

AN FSP FRAGMENT IN WHICH DATA IS FILTERED THROUGH A $TEST$ PROCESS.

action that causes different actions to be performed depending on the outcome of the test.

The FSP fragment in Table II implements a test by augmenting the process associated with a $store$ action. As usual, a process action p_1 is synchronized with a store process through the action end_p_1. Inserted into the store process is $TEST\_t_1$, consisting of the branching staging action labeled by the transitions yes, no that conduct the test. In this example, the trigger is true (yes) if the image is considered to be cloud-free (e.g. indicating that the image is a certain percentage clear).

E. Summary of model components

This completes the presentation of finite state model for sensor web reconfiguration. The semantics of workflows can be summarized by a small set of primitive notions:

- Basic actions come in two types: WA actions and staging actions;
- WA actions include acquiring, processing and storing data;
- WA actions are organized into processes with staging actions interleaved with WA actions in sequence or conditionally through branching;
- A workflow is a set of processes executing in parallel, with ordering accomplished through synchronization.

One advantage of the finite state model is that it can provide the foundation for automating the execution of a workflow. The next section describes a model for execution based on the model introduced in this section.

V. Workflow Execution

Executing a workflow autonomously is facilitated by transforming a workflow specification into a form that can be interpreted by a software agent (which we call a *web manager*) as a sequence of commands to reconfigure the sensor web. The semantics of the LTS model provides the basis for this transformation. The web manager acts as a controller, a device for determining the current state of the system, executing the actions on the labeled transitions from that state, and determining the new state that results.

The main loop of the web manager algorithm can be represented as follows:

Loop
    S = InitialState
    While ($S \neq STOP$)
        $(a \rightarrow Q) = SelectTransition(S)$
        DoAction(a)
        $S = Q$
end loop

The web manager reads the current state, S, selects a transition out of the state, does the action required of the transition, and resets the current state to the state determined by the chosen transition.

A. Selecting a Transition

Given a state s in an LTS, let $T(s)$ be the set $\{a_1 \rightarrow Q_{a_1}, \ldots, a_n \rightarrow Q_{a_n}\}$ of labeled transitions out of s, where the a_i are actions and the Q_{a_i} are states. For example, in Figure 3, $T(3) = \{done \rightarrow 4, not\_done \rightarrow 0\}$. Given a current state S, the web manager will select one among the actions a_i such that $a_i \rightarrow Q_{a_i} \in T(S)$. Clearly, if $T(S)$ consists of a single transition, the web manager will select that transition.

There are potentially many different cases to consider in the general case, i.e., where $T(S)$ consists of a set of n transitions. In the current framework, based on the scenarios we've been exploring, it is possible to assume that $T(S)$ consists of one of the following:

- A single transition;
- A set of n transitions, where for each $a_i \rightarrow Q \in T(S)$, a is an *acquire data* action;
- A set of two elements $a_1 \rightarrow Q_{a_1}, a_2 \rightarrow Q_{a_2}$, where a_1 and a_2 are mutually exclusive staging actions, i.e., where the corresponding triggers are of the form $E, \neg E$.

This means, for example, that no mixture of transitions with WA actions and staging actions occur. Clearly, if there are cases that don't fit into one of the three just listed, they can be easily accommodated.

Consequently, the algorithm for selecting a transition can be represented as follows:

$SelectTransition(S : State)$
$T = T(S)$
if $T = \{a \rightarrow Q\}$
    return $a \rightarrow Q$
else if for some transition $a \rightarrow Q \in T$, a is acquire data
    non deterministically choose one of the transitions in T
    return $a \rightarrow Q$
else $T = \{a_1 \rightarrow Q_{a_1}, a_2 \rightarrow Q_{a_2}\}$.
    Let t be the trigger of a_1.
    If t is true, then return $a_1 \rightarrow Q_{a_1}$,
    else return $a_2 \rightarrow Q_{a_2}$

In the second case, non-deterministic choice is an option because the order in which the acquire data actions are executed does not matter. In particular, this instance occurs when a process action requires more than one acquire data actions as inputs.

B. Executing Actions

Executing a basic action in the context of sensor web reconfiguration means either sending a request to reconfigure the web or observing some change in the reconfiguration of the web. Thus, the action being executed can be said to be *controllable* or *uncontrollable* in the sense of [9]: roughly, a controllable action is one the agent (in this case, the web manager) executes directly, whereas an uncontrollable action is one that is executed by the "system" (i.e. the sensor web itself) to indicate progress in the process of reconfiguration.

If the action to be executed is a WA action, then the action is transformed into an executable *request*, described in more detail below. Otherwise, the action is a staging action. Notice that some staging actions don't require sensor web access: such actions have triggers that can be determined 'locally' by the web manager. Notice that this kind of action is executed in the select transition phase, and does not need to be executed again.

The interesting case with staging actions is where a trigger is something that needs to be observed (in particular, that a WA action has been completed). This happens when the staging action is a synchronizing action. In this case, the web manager may need to delay the execution process until the trigger has been observed.

Thus, the algorithm for executing actions can be stated as follows:

$DoAction(a : Action)$
if a is a WA action
    $submitRequest(a)$
else let t be the trigger of a
    $waitForTrue(t)$

C. Submitting Requests and Execution Monitoring

The process of executing an action involves transforming a basic action into either a sensor web request or other control action. A sensor web request is an atomic action (at least from the standpoint of the web manager) for performing a reconfiguration of the sensor web. This transformation requires

two sources of inputs: inputs provided by the user and inputs generated internally by the system as part of the service layer discussed earlier. Collectively these inputs provide the content and means to encode an *executable request* to task a sensor, retrieve data from an archive, run a processor of data, or store some result.

At a high level of abstraction we can view the inputs to the transformation process as involving an *action argument table* (AAT) and a *procedure map* (PM). As the name suggests, the AAT holds, for each action, the list of values for arguments that are required to execute the action. For each WA action, the table entry will contain values for location, time, sensor, and other values related to quality of the acquisition. Each staging action will contain an AAT entry for the trigger associated with the action.

A PM can be viewed as a function that maps a WA action and its arguments into executable sensor web requests. Minimally, the procedure map contains, for every sensor and measurement type, an *address (path)* and *query template* for formatting and submitting a request to the sensor web. Figure 4 illustrates the transformation of a basic action using the AAT and Procedure map. It should be stressed that this is a schematic representation of the transformation process, which may in fact involve a set of processing steps. Furthermore, as suggested in Figure 2 above, the transformation process from WA action to sensor web request could involve intermediate requests to a service layer for information that is required for sensor web access.

VI. Extensions

A. Failure Management

The existence of uncontrollable actions means that an action may fail because of something happening unexpectedly. The LTS model presented here lends itself to extensions for enabling *recovery actions* in the face of unexpected failure in reconfiguration actions. Recovery strategies can be simple (e.g. *fail-safe*) or complex. An example of the former is suspending the entire workflow if one action fails. For example, if a request acknowledgment is not received, this strategy would simply transition into a special termination state indicating failure. Other strategies may involve retrying an action on failure: for example, resubmitting the data acquisition request if the timed out indicator has been surpassed. Still other strategies may involve sending an alert to the scientist.

B. Automated goal generation

The campaign cycle is renewed when the data from observations are analyzed. New science goals can be viewed very broadly as arising from the perception of gaps or discrepancies between model predictions and observations. This paper has not considered the problem of automating this phase of the process. Other work [8] has examined the use of data mining and machine learning to automatically identify possible errors in model predictions with respect to observation made during the INTEX-B mission described earlier. The locations where the errors occur are transformed into new mission goals for DC-8 flight plans for the subsequent cycle.

VII. Relevant research

The work described here is based on, or has similarities to, other work in different areas of planning and control. This section offers a sample of related work.

A. Workflow and Dataflow Planning

Workflow management is a process of defining, managing and executing workflows, sets of procedures for passing files and data in order to accomplish a goal. The layers of a typical workflow management system [11] are similar to those of a system for managing a sensor web, from application modeling, to workflow specification and workflow execution. indeed, the sensors that make up a sensor web can be viewed as special kind of grid resources. An previous attempt to develop a planner-based system for generating and executing data-flow procedures to generate requested data products is found in [5].

B. Service Oriented Multi-agent systems

Service-oriented computing [6] (SOC) is the use of computational resources in order to automatically execute a service (e.g. shipping, travel, consumer goods purchasing). The scope of SOC is expanding due to the increasing need for ways to discover and coordinate services. One of the research areas responding to the expansion of scope is research in Multi-Agent Systems (MAS). One component of this response is the idea that services are themselves ”agents” that automatically manage the details in execution the service, allowing the users to focus on high-level abstractions. An instantiation of this approach is found in [3].

C. Model-based Autonomous Control

Model-based autonomy [10] is the paradigm of creating intelligent systems that automatically plan courses of action and diagnose their state. Part of this paradigm involves programming such systems to achieve desired states by using a model to capture the system’s nominal behavior and common failure modes. A model-based program is executed by automatically generating a sequence of control actions that moves the system into its desired state. The work described here follows this basic paradigm of programming and executing control sequences based on a model.

D. Standards and Infrastructure

The work described here complements and expands efforts, such as OGC-SWE [2], to develop standards for discovering and acquiring observations, as well as tasking sensors. SWE provides four web services: observation (Sensor Observation Service), alert (Sensor Alert Service), planning (Sensor Planning Service) and notification (Web Notification Service). Discovery of services is facilitated by the Sensor Web Registry Service.

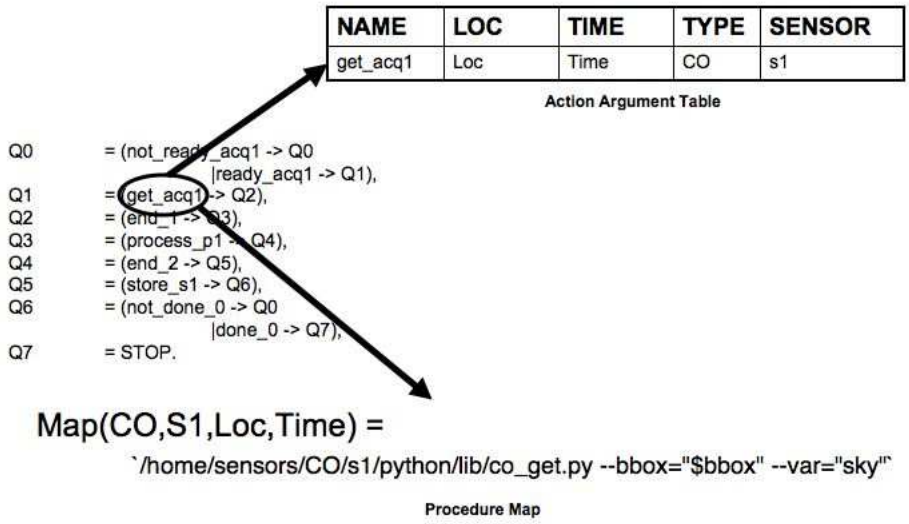

Fig. 4. A schematic representation of the process of transforming a basic data acquisition action into an executable sensor web request.

VIII. Conclusion

This paper has introduced a model of sensor web configuration based on principles underlying planning and finite state control. Planning involves the transformation of goals into executable sequences of actions. Control involves procedures for enabling the safe transition of a controlled system into a desired state. The representation of a workflow as a labeled transition system links planning with control, and realizes the notion of sensor web management as a goal-directed process.

References

[1] J. Magee andJ. Kramer. *Concurrency: State Models and Java Programming*. Wiley, 2006.

[2] Michael Botts. Ogc sensor web enablement: Overview and high level architecture. In *Open GIS White Paper OGC 06-050r2*, 2006.

[3] Moodley D., Terhorst A., Simonis I., Mcferren G., and Van den Bergh F. Using the sensor web to detect and monitor the spread of wild fires. In *2nd International symposium on geo-information for disaster management*, Goa, India, 2006.

[4] Christian Frank, Vlado Handziski, and Holger Karl. Service discovery in wireless sensor networks. URL: citeseer.ist.psu.edu/frank04service.html.

[5] K. Golden, Wanlin Pang, Ramakrishna Nemani, and Petr Votava. Automating the processing of earth observation data. In *Proc. Int'l Symp. AI, Robotics, and Automation in Space*, 2003.

[6] Michael N. Huhns, Munindar P. Singh, Mark Burstein, Keith Decker, Edmund Durfee, Tim Finin, Les Gasser, Hrishikesh Goradia, Nick Jennings, Kiran Lakkaraju, Hideyuki Nakashima, H. Van Dyke Parunak, Jeffrey S. Rosenschein, Alicia Ruvinsky, Gita Sukthankar, Samarth Swarup, Katia Sycara, Milind Tambe, Tom Wagner, and Laura Zavala. Research directions for service-oriented multiagent systems. *IEEE Internet Computing*, 9(6):65–70, November 2005.

[7] INTEX-B. http://www.espo.nasa.gov/intex-b/.

[8] Elif Kürklü, Robert A. Morris, and Nikunj Oza. Learning points of interest for observation flight planning optimization: A preliminary report. In *Proceedings of the Workshop on AI Planning and Learning*. International Conference on Automated Planning and Scheduling, 2007.

[9] P. Morris and N. Muscettola. Execution of temporal plans with uncertainty. In *Proc. of AAAI-2000*, pages 491–496, 2000.

[10] B. Williams, M. Ingham, S. Chung, and P. Elliott. Model-based programming of intelligent embedded systems and robotic space explorers. *Proceedings of the IEEE: Special Issue on Modeling and Design of Embedded Software*, 91(1):212–237, 2003. Invited Paper.

[11] J. Yu and R. Buyya. A taxonomy of workflow management systems for grid computing, 2005.